Gm327

THE
MULTINET
OEM
MANUAL

INTRODUCTION

The MultiNet OEM disk contains a set of routines which provide facilities for sending and receiving blocks of data from different stations connected to a MultiNet network. The routines are stored as Microsoft format REL files, so to do this the routines must be linked into the application program using a Microsoft compatible linker (eg. Microsoft's LINK-80 (L80)).

The disk provided contains two files, MASTER.REL and SLAVE.REL. These are the MultiNet software components for the master and slave stations respectively. The files may be linked into the application program, and they provide four routines which are declared public and so may be called from within the application program. These routines have the labels INIT, ADDRESS, PKTOUT and PKTIN and their use is described elsewhere in this manual.

The MultiNet software must be run in RAM to operate, so if the application concerned is ROM based, then the MultiNet software must be copied across into RAM before being executed. Since much of the timing of network operation is performed in software, the software must only be run on a 4MHz system **without** wait states.

The network software expects the PIO connecting it to the network hardware (GM836) to be located at port OB4H onwards. If the application software is to be run under a Gemini CP/M system then the Centronics parallel port assignment will have to be altered to avoid conflict with this port. This is done by setting the Centronics port either to an alternative address, such as that of a PIO on the Gemini GM816 I/O board, or to zero, as described in the appropriate manual for the system. If the program CONFIG.COM is available this may be used.

When executing, the MultiNet software must not be interrupted, since it is a real-time system and any interrupts will result in loss of data.

The network software does not alter the Z80 alternate register set, so these may be used by the application program without having to save them on each call to a network function. However, the MultiNet software does not create its own stack, but makes use of the application program's stack. Thus, the application program should provide adequate stack space and at least 8 levels must be allowed for this purpose.

INFORMATION TRANSFER

The network operates by having a single master station poll several slave stations. Because of the way the network protocol is designed, the format of data transfer **must always** be a transmission of data from a slave to the master (the particular slave concerned being selected by the master) followed by a reply consisting of a transmission of data from the master to the slave. This reply may be a dummy data block if nothing actually needs to be sent. Similarly, if no data needs to be transferred from the slave to the master then a dummy block may be sent. This dummy block may either be a zero length block or it may also be marked by a special packet mode (see later) to indicate dummy data. The master-slave polling mechanism is built in to the MultiNet software and is transparent to the programmer apart from the restrictions described above. It is not possible for slaves to transmit any information until requested to do so by the master since they will wait until polled by the master (i.e. until the master is ready and willing to receive the data).

The basic unit of data which is sent through the network is the block. This is a contiguous block of memory which may have a length of between 0 and 255 bytes. Generally, the function routines use register HL as a pointer to the start of this block, either for transmitting the block or for receiving a block at (HL). Register B is used to hold the length of the data block and so may take the value 0 to 255.

A secondary unit of information which may be transmitted is the packet mode. This is a single byte of information which is sent along with the data block, and is normally used to indicate the type of data which is being sent (for example urgent data, normal data, self-test information, dummy block etc.). Its value is not used by the network software at all, and so it may be used for any convenient purpose by the programmer. The packet mode is sent/received in register C. The packet mode information is sent even if the block length is zero. Thus, sending a zero length block results in only the packet mode (1 byte) being transferred from register C in one machine to register C in the other machine.

The master system also has to specify a wait count for a block of data when it is sent/received. This is the timeout count which determines how long the master will wait for a positive response to a poll of the slave station before aborting and indicating an error status. The higher the value of the wait count which is held in register DE, the longer the master will wait (although 0 actually gives a longer wait than FFFFH), with the minimum practical value being about 10H. Values smaller than this result in wait times which are too short to overcome the internal delays inherent in the network. The value of the wait count will depend on the network application, and will be different for sending and receiving. On receiving data from a slave into the master, the wait count determines how long the master will wait for the slave to send data in response to the master's poll. On sending data from the master to the slave the wait count will determine how long the master is prepared to wait for the slave to become ready to receive that data.

Two of the four routines are different for master and slave systems, while the other two are common to both the master and slave systems. Detailed information on these now follows.

****
***** INITIALISATION *****
****

Routine name:
INIT

On entry:
(No parameters)

On exit:
(No returned values)

The INIT routine is used to perform initialisation of the GM836 hardware and it must be called before any use is made of the network.

****
***** RETURN BUS ADDRESS *****
****

Routine name:
ADDRESS

On entry:
(No parameters)

On exit:
A = Bus address

This routine is used to return the physical station address of the GM836 network board. This is the address set by the 5-way DIL switch on the board in the range 0 - 31 and is unique for each of the stations attached to the network.

For the MASTER system:

```
***********************
***** SEND PACKET *****
***********************
```

Routine name:
PKTOUT

On entry:
A = Destination address
B = Length of data block
C = Packet mode
DE = Wait count
HL = Data block start address

On exit:
A = Return code
B, C, D, E, H, L Undefined
Z = Block transmitted
NZ = Block not transmitted

The send routine is used to send a block of data from the master station to a slave station. This must only be done as the reply to a block of data received from a slave station. Register A contains the bus address of the destination slave station (which will be the same as that returned by the previous receive function call). An error status will also be returned if the slave station did not send any data within the time determined by the wait count.

For the MASTER system:

```
***************************
***** RECEIVE PACKET *****
***************************
```

Routine name:
PKTIN

On entry:
A = Address of source
B = Maximum data length
DE = Packet wait count
HL = Block start address

On exit:
A = Return code
B = Number of bytes received
C = Packet mode
D, E, H, L Undefined
Z = Packet received correctly
NZ = Packet received in error

The receive routine is used to obtain a block of data from a slave station. Register A contains the bus address of the transmitting slave station. Register B is initially set to the maximum buffer length available and upon return from the routine is set to the number of bytes actually received. If an attempt is made to overflow the buffer (i.e. the slave station sends a data block which is too long) then an error status is returned. An error status will also be returned if the slave station does not reply to the poll within the wait time specifed in DE.

For any of the SLAVE systems:

```
***********************
***** SEND PACKET *****
***********************
```

Routine name:
    PKTOUT

On entry:
    B = Length of data block
    C = Packet mode
    HL = Data block start address

On exit:
    A = Return code
    B, C, D, E, H, L Undefined
    Z = Block transmitted
    NZ = Block not transmitted

The send routine is used to send a block of data from a slave station to the master station. The routine will not send the data block until requested to do so by the master.

For any of the SLAVE systems:

```
*************************
***** RECEIVE PACKET ****
*************************
```

Routine name:
    PKTIN

On entry:
    B = Maximum data length
    HL = Block start address

On exit:
    A = Return code
    B = Number of bytes received
    C = Packet mode
    D, E, H, L Undefined
    Z = Packet received correctly
    NZ = Packet received in error

The receive routine is used to obtain a block of data from the master station. Register B is initially set to the maximum buffer length available and upon return from the routine is set to the number of bytes actually received. If an attempt is made to overflow the buffer (i.e. the master station sends a data block which is too long) then an error status is returned.

ERROR HANDLING

The function routines return an error status using the Z flag upon completion. If Z is set then the function was performed correctly, whereas if Z is reset (NZ) then an error occurred during the operation and in this case a return code is held in register A which gives the cause of the error. Most of these error codes are meaningless without knowing the method of operation of the software, (error codes 2, 3, 6, 7, 9) but three (error codes 1, 4, 10) are useful to the programmer.

Error 1. No block sent. This error occurs when the master requests a block of data from a slave using the PKTIN routine and the data is not sent by the slave in the time specified by the wait count in DE. This will occur if the slave station does not wish to transmit any data, and in this case the master may choose to poll another slave station, or it may try to poll the same slave again. The error may also occur if the wait count in DE is so small that the slave cannot reply quickly enough before the master gives up waiting. If this is the problem then the solution is to make the wait count higher. Wait counts below about 10H are too small and can cause this problem. This error can only occur on the master system.

Error 4. Data block too long for specified buffer. This indicates that the maximum data block (buffer) length specifed in register B for a receive function call was too small to contain the data block being received. In this case no data is put into the buffer at (HL).

Error 10. No reply to master poll. If a slave station has sent a data block to the master and does not then wait for a reply from the master then this error will be generated. If this error occurs the most likely cause is that the wait count on the master PKTOUT routine is set too short, so it does not wait for long enough for the slave to become ready to receive the reply. This error can only occur on the master system.

Any error return other than the three above indicates that a non-recoverable netowrk error has occurred which the network software cannot correct.

SIMPLE SOFTWARE EXAMPLE

The following two programs form a skeletal network interface and represent the minimum software required in an application. They may be used purely as examples of the software required, or they may be run under a debugger such as ZSID, or GEMDEBUG to check out hardware and gain familiarity with the system. If running under a debugger, note that the MultiNet software must not be executed in trace mode as this will not run it at its proper speed and timing errors will result. In this case the calls to the MultiNet software must be executed directly and any tracing restricted only to the software below.
Also note that the SLAVE equate in the software for the master must be set to the bus address of the slave station with which you wish to communicate (in the program it is set to station 9).

```
        TITLE Test Program for MultiNet Slave

        .Z80

;Network functions

EXTRN   INIT,PKTOUT,PKTIN

;Send data to master

        LD      SP,STACK        ;Set up stack
        CALL    INIT            ;Initialise network
        LD      B,OFFH          ;Send 255 bytes
        LD      C,"T"           ;Packet mode is "Test"
        LD      HL,XMITBUF      ;Send from transmit buffer
        CALL    PKTOUT          ;Send the data

;Receive reply from master

        LD      B,OFFH          ;Get up to 255 bytes
        LD      HL,RCVEBUF      ;..into receive buffer
        CALL    PKTIN           ;Wait for the data

;Workspace

XMITBUF:DEFS    255             ;Transmit buffer
RCVEBUF:DEFS    255             ;Receive buffer

        DEFS    32              ;Stack space
STACK:

        END
```

```
        TITLE Test Program for MultiNet Master

        .Z80

;Slave station with which to communicate

SLAVE   EQU     9               ;Use station 9

;Network functions

EXTRN   INIT,PKTOUT,PKTIN

;Receive data from slave

        LD      SP,STACK        ;Set up stack
        CALL    INIT            ;Initialise network
        LD      A,SLAVE         ;Select slave station
        LD      B,OFFH          ;Get up to 255 bytes
        LD      HL,RCVEBUF      ;..into receive buffer
        LD      DE,8000H        ;Wait about 1 sec.
        CALL    PKTIN           ;Get the data

;               .               (Process data and
;               .               ..prepare reply)

;Send reply to slave

        LD      A,SLAVE         ;Select slave for reply
        LD      B,OFFH          ;Send 255 bytes
        LD      C,"T"           ;Packet mode is "Test"
        LD      HL,XMITBUF      ;Send from transmit buffer
        LD      DE,8000H        ;Wait about 1 sec.
        CALL    PKTOUT          ;Send the data

;Workspace

XMITBUF:DEFS    255             ;Transmit buffer
RCVEBUF:DEFS    255             ;Receive buffer

        DEFS    32              ;Stack space
STACK:

        END
```

EXAMPLE OF MASTER SYSTEM INTERFACE SOFTWARE

This example shows how polling is achieved on the master system. The example implements a two level scheduling system where only the active slave stations are polled frequently but there is a global poll occasionally to see if there are any more stations wishing to transmit. The actual details of this are largely irrelevent but are included to give a wider view of the software interface.

```
;*********************
;***** SCHEDULER *****
;*********************

;Do round robin poll of all possible stations

RR:     LD      A,(RRCNT)
        DEC     A               ;New station address
        LD      (RRCNT),A
        LD      (STATION),A
        JR      NZ,POLL         ;Poll if not all done
        LD      A,32            ;Else set count
        LD      (RRCNT),A
        LD      HL,0            ;..and poll count
        LD      (POLLCNT),HL    ;Fall through to scheduler

;Check whether to despool a character

SCHED:  LD      A,(PRTCNT)      ;Get print count
        DEC     A               ;..and update
        JR      NZ,NEWPRT
        LD      A,20            ;Print once every 20 polls
NEWPRT: LD      (PRTCNT),A      ;Resave count
        CALL    Z,DESPOOL       ;Despool a character

;Schedule next function call

        LD      HL,(POLLCNT)    ;Time to look at everyone?
        DEC     HL
        LD      A,H
        OR      L
        JR      Z,RR            ;Yes, schedule all round robin
        LD      (POLLCNT),HL    ;Else update count
        LD      A,(STATION)     ;Get station number
        CP      32              ;Update station number
        JR      NZ,SCHED1
        XOR     A
SCHED1: INC     A
        LD      (STATION),A     ;Save it
        LD      E,A             ;Convert to 16 bits
        LD      D,0
```

```
        LD      HL,USRTAB       ;Get current station's
        ADD     HL,DE           ;..vector entry address
        LD      A,(HL)          ;Get station status
        OR      A               ;(Zero means not logged on)
        JR      Z,SCHED         ;Again if not logged on
        LD      A,E             ;Get back station number

;Poll for reply from station

POLL:   LD      B,255           ;Maximum packet length
        LD      HL,MSGBUF       ;Buffer address
        LD      DE,10H          ;Wait count
        CALL    PKTIN           ;Get the packet
        JR      NZ,SCHED        ;Try again if nothing

;**************************
;***** ACCESS MANAGER *****
;**************************

;Set user number from station number

        LD      A,(STATION)     ;Get station number
        LD      E,A             ;And make it 16 bits
        LD      D,0
        LD      HL,USRTAB       ;Start of table
        ADD     HL,DE           ;Index station number
        LD      A,(HL)          ;..to get user number

                .
                .         (Process packet received)
                .         (Set B to length of reply)
                .         (Set A to return status)
                .

;Send back parameters to station being serviced

RTN:    LD      HL,MSGBUF       ;Start of buffer
RTN1:   LD      C,A             ;Save return code
RTN2:   LD      A,(STATION)     ;Station address
        LD      DE,800H         ;Wait count
        CALL    PKTOUT          ;Send it
        JP      SCHED           ;Loop forever
```

EXAMPLE OF SLAVE STATION INTERFACE SOFTWARE

This example gives the corresponding slave station software to that for the master in the previous section. In the example, the routines FNOUT and FNIN are used to send and receive data. The slave station must always output data in response to a poll from the master and then wait for a reply (in this case containing a return status, although it may be a dummy reply). They are used in the form:

```
        LD      B,33            ;FCB length
        EX      DE,HL           ;FCB address in HL
        PUSH    HL              ;Save it
        CALL    FNOUT           ;Send packet
        LD      B,33            ;Want an FCB back
        POP     HL              ;..at this address
        CALL    FNIN            ;Get reply
        RET
```

The routines are as follows:

```
;Send function call to server

FNOUT:  CALL    PKTOUT
        RET     Z               ;Go back if ok
        LD      D,A
        LD      E,"T"
        JR      ERRDEC          ;Else error

;Wait for reply to function call

FNIN:   CALL    PKTIN
        LD      D,A
        LD      E,"R"
        JR      NZ,ERRDEC       ;Skip if network error
        LD      A,C             ;CP/M return code in A
        CP      OFFH            ;CP/M return code?
        RET     Z
        CP      80H
        RET     C
ERRDEC: LD      A,D
        ADD     A,'0'-1         ;Scale '0' - '9'
        LD      D,A
        LD      (ERRNMR),DE
        LD      DE,NETERR       ;Fall through to print it
```

CABLING CONSIDERATIONS

The MultiNet twisted pair cable should be carefully sited. Running the cable through electrically 'noisy' environments and strong magnetic fields (such as those produced by large transformers) should be avoided. Electrical 'noise' is caused by heavy equipment operating, (X-ray machines are particularly notorious for this), and whilst MultiNet is fairly tolerant to this type of interference, it makes sense to spend a little time planning the layout of the wiring to avoid any potential problems.

The cabling necessary to connect all the workstations to the server takes the form of a single length of twisted pair, to which all the machines are attached, as shown below.

```
                                        -------             -------
                                        |     |             |     |
                                        -------             -------
                                           |                   |
   [ ]--------------[ ]-----------[ ]-----------[ ]------[ ]
    |                 |                          |
 -------           -------                    -------
 |     |           |     |                    |     |
 -------           -------                    -------
```

Machines are attached to the main twisted pair bus by means of junction boxes represented above by '[ ]'. The junction boxes have a three pin connector fitted which mates with a corresponding connector on the short length of twisted pair between the machine and its junction box (known as a 'spur').

As can be seen from the diagram above the layout of cable is very simple, but there are several points which **MUST** be adhered to, to give a reliable system.

1. Although cable may be 'snaked' around as much as is necessary, there must be **NO** loops of any sort in the cabling. In particular, note that the ends of the cable are not joined together.

2. The ends of the cable must be terminated to avoid the possibility of reflections. 120R resistors are used for this purpose, fitted into the two junction boxes at the extreme ends of the cable. Terminating resistors must not be fitted anywhere else along the length of cable.

2. The length of spurs used should be kept as short as is practically possible. They should not be longer than 3m (10').

3. The shield of the twisted pair cable should be connected to ground at one point only, or earth loops may result. This point is normally at the master system. The junction boxes preserve continuity of the shield and also provide a shield connection to the spur, right up to the connector on the workstation. However, to fulfil the requirement of a single ground point, the slave stations must not ground the shield, but must leave it unconnected.

4. The overall length of the cable used should not exceed 600m (2000'). In practice this should not be a problem, since most installations will, in fact, use much less than this amount.

CONNECTION OF JUNCTION BOXES

The junction boxes supplied may be mounted in any convenient place, bearing in mind the restrictions mentioned in the previous section. Wiring the junction box is relatively straightforward operation and should present no problems.

To connect a junction box to the twisted pair cable the following steps should be followed:

1. Cut the twisted pair cable at the point where the junction box is to be fitted, and dismantle the two halves of the junction box.

2. Make sure the cable is passed through the appropriate guide holes in the junction box before the two pieces of cable are joined together again!

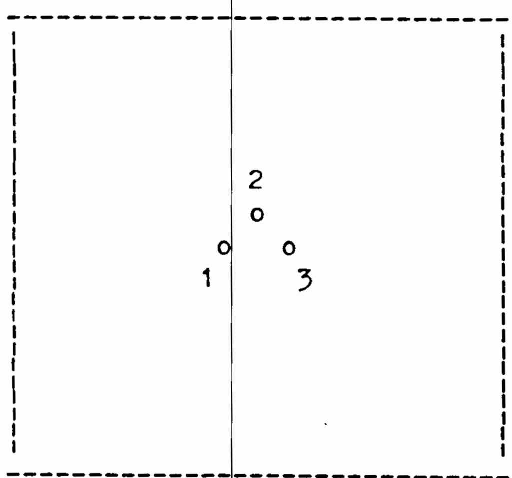
3. Strip the outer PVC insulation, and tear off the foil shield for a length of about 2 cm from both of the free ends of the cable.

4. Strip the inner PVC insulation of both conductors of the twisted pair, to a length of about 0.5 cm, from both free ends of the cable.

5. Tin both conductors of the twisted pair, and also the uninsulated drain wire which carries the ground connection, again do this for both ends.

6. Twist the corresponding conductors together, to once again form one continuous length of cable (ie. twist black & black, white & white, and the two drain conductors together).

7. Solder the three conductors to the three pins of the connector on the junction box. The pinout of the connector is as shown below, **viewed from behind.**



Pin 1 = Black conductor
Pin 2 = Drain conductor
Pin 3 = White conductor

8. If the junction box is one of the two at the extreme end of the cable, then a terminating resistor must be fitted. To do this, solder a 120R resistor across the two twisted pair conductors, where they are soldered to the connector. This is between pins 1 and 3.

9. Before reassembling the junction box, check that the joints are well soldered and that there are no short circuits between conductors.

When all the junction boxes in the system have been connected check for continuity between the two extreme ends of the cable. There should be continuity between equivalent pins on the junction boxes. If there is an open circuit, then it is a simple matter to trace along the junction boxes on the cable to find the point where the fault lies. Also, check for short circuits between the pins on one of the junction boxes. If there is a short circuit somewhere, it will be a matter of inspecting each junction box to find it.

STATION CONFIGURATION

The interface between a machine and the network takes the form of the GM836 board which plugs into the PIO on the CPU board of the machine and a 26-way ribbon cable is used for this purpose. To connect the machine to the network, all that it is necessary to do is to connect the free end of the cable forming a spur from a junction box to the connector on the network interface board.

One other thing must be done, however, before the network can be used, and that is the station address of the machine must be set. This is done by setting the address on a 5-way DIL switch on the network interface board. Up to 31 slaves plus a single master may be connected to the network, although in practice this number would usually be a lot less depending on the exact usage of the installation.

To set the station address first decide what address is required. Any station address may be used providing it is not in use by another station, every station must have a unique station number. In addition, the master system is assigned the address zero by convention. Once you have decided on the station number, look at the table below and set the 5 positions of the DIL switch to the values shown in the table. U = up, and D = down with the workstation in the orientation with the cable end of the network board lowermost. The pattern set on the DIL switches is that shown below when looking at the network interface board from the side with the silk-screening.